APPENDIX IV - ERROR HANDLING APPLICATION


```
public class ErrorHandlingAppSample implements IEventHandler
{
  private final static int MAX_EVENT_STORE = 5;
  private final static int ID_FOR_APP_SAMPLE = 55;   // typically set by the system
  private static int eventCount = 0;
  private IMessageEvent [ ] imeStore = new IMessageEvent[MAX_EVENT_STORE];

  /**
   * The zero argument constructor demonstrates a possible application example where
   * the application registers to receive error events, logs events, and registers to
   * receive reboot events. The SysSample class contains that code that will generate
   * a sample reboot event.
   */
  public ErrorHandlingAppSample( )
  {
    // Get the default system error handler registrar.
    SysHandlerRegistrar ehr =
        SysHandlerRegistrar.getInstance( );

    // Set this object as the new error handler.
    ehr.setEventHandler(SysHandlerRegistrar.ERROR_INFO_EVENT_HANDLER, this);
  }

  /**
   * Receive a message event from the EventProcessor. This method will be used to process
   * all of the error and informational messages sent to the registered error handler and
   * other applications. This sample simply places the messages into an array. Additional
   * processing is specific to the application. For example, an application may look at
   * the error code and application identifier of the event and take recovery action for
   * specific errors. In case of a critical error the handler may send a message to a
   * server agent.
   *
   * @param see - Event generated by the system or sent by an application.
   *
   * @return The event unchanged, or the event modified to suit the purposes of the
   * registered registered event handler, or null to indicate that the registered handler
   * has consumed the event.
   */
  public IMessageEvent receiveEvent(IMessageEvent see)
  {
    System.out.print("ErrorHandlingAppSample.receiveEvent(); event type: ");
    System.out.print(see.getTypeCode());
    System.out.print("; date: ");
    System.out.println(see.getDate());

    eventCount = (eventCount == MAX_EVENT_STORE - 1) ? 0 : eventCount + 1;

    imeStore[eventCount] = see;    // Store the event for later retrieval.

    return null;      // Tell the EventDatabase that the registered handler has consumed
                      // the event.
  }
```

```
    /**
     * Get any events saved by the handler.  A network server agent may poll a client agent
     * running in the same device as this handler so that the client agent can get the
     * events using this method.
     *
     * @return The array of events or null if none were stored.
     */
    public IMessageEvent [] getEvents()
    {
      return imeStore;
    }

}
```